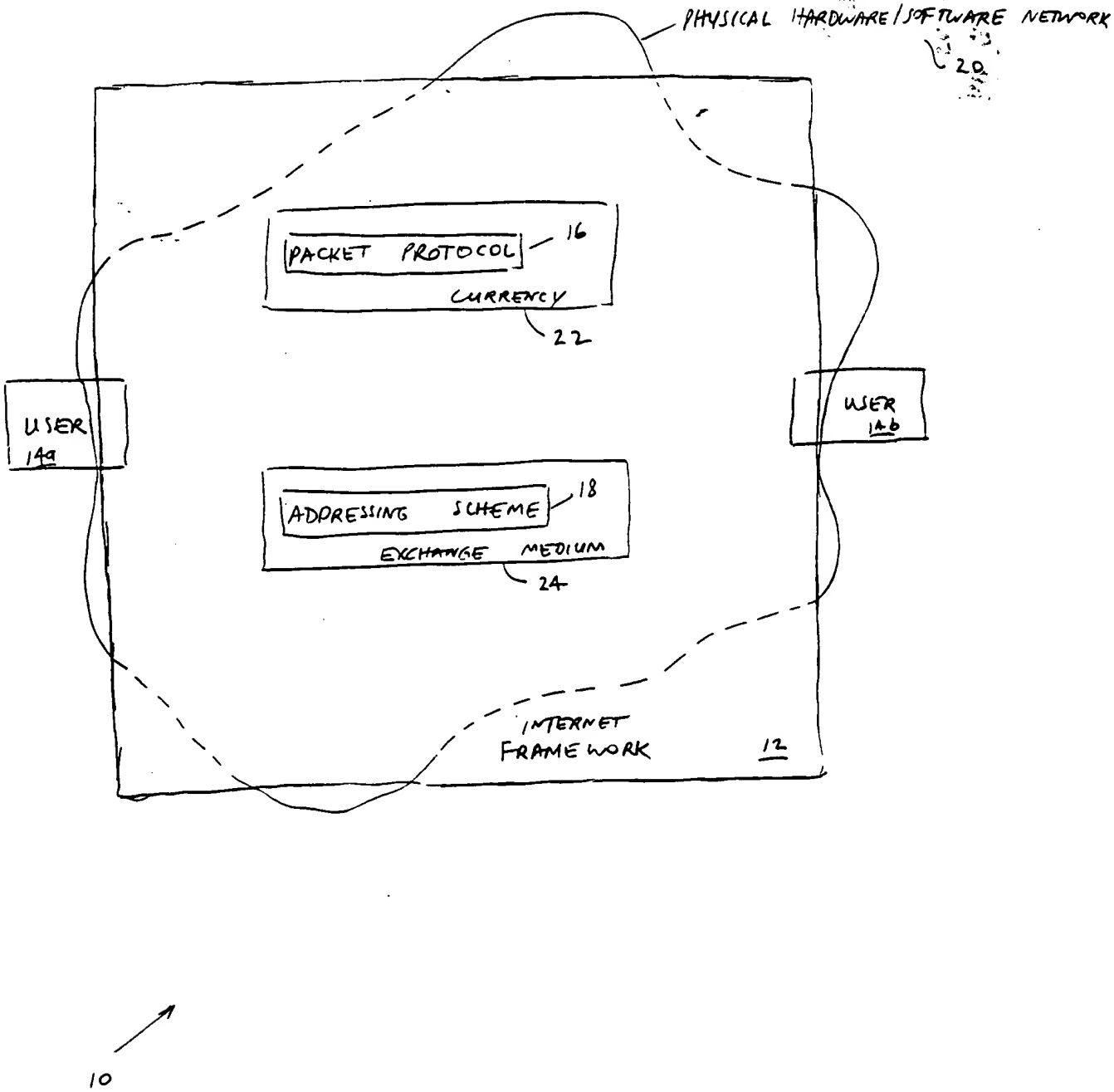

PHYSICAL HARDWARE/SOFTWARE NETWORK
20
PACKET PROTOCOL
16
CURRENCY
22
USER
14a
USER
14b
ADDRESSING SCHEME
18
EXCHANGE MEDIUM
24
INTERNET
FRAME WORK
12
10

FIGURE 1

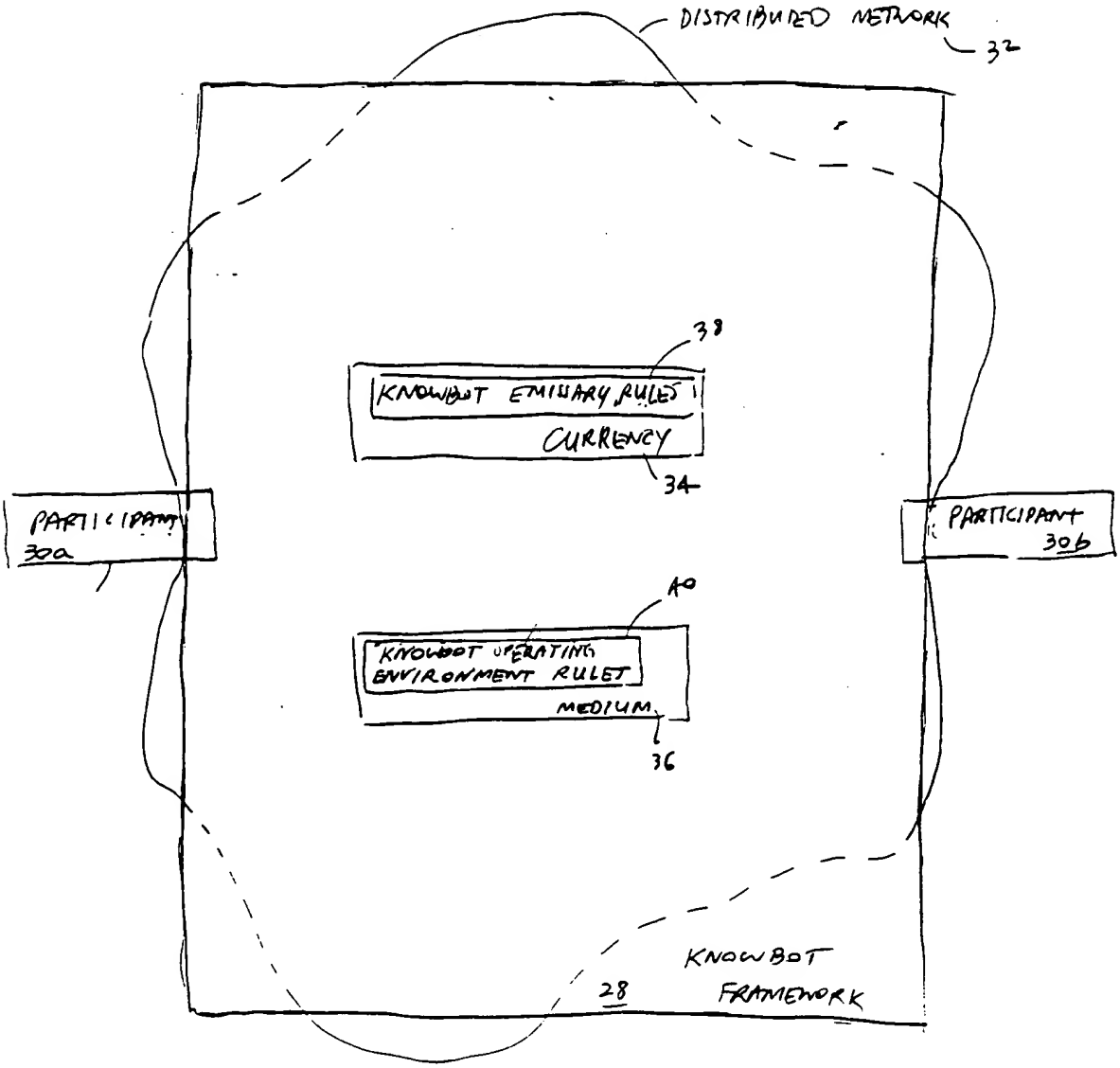


FIGURE 2

95/25850 26002.80

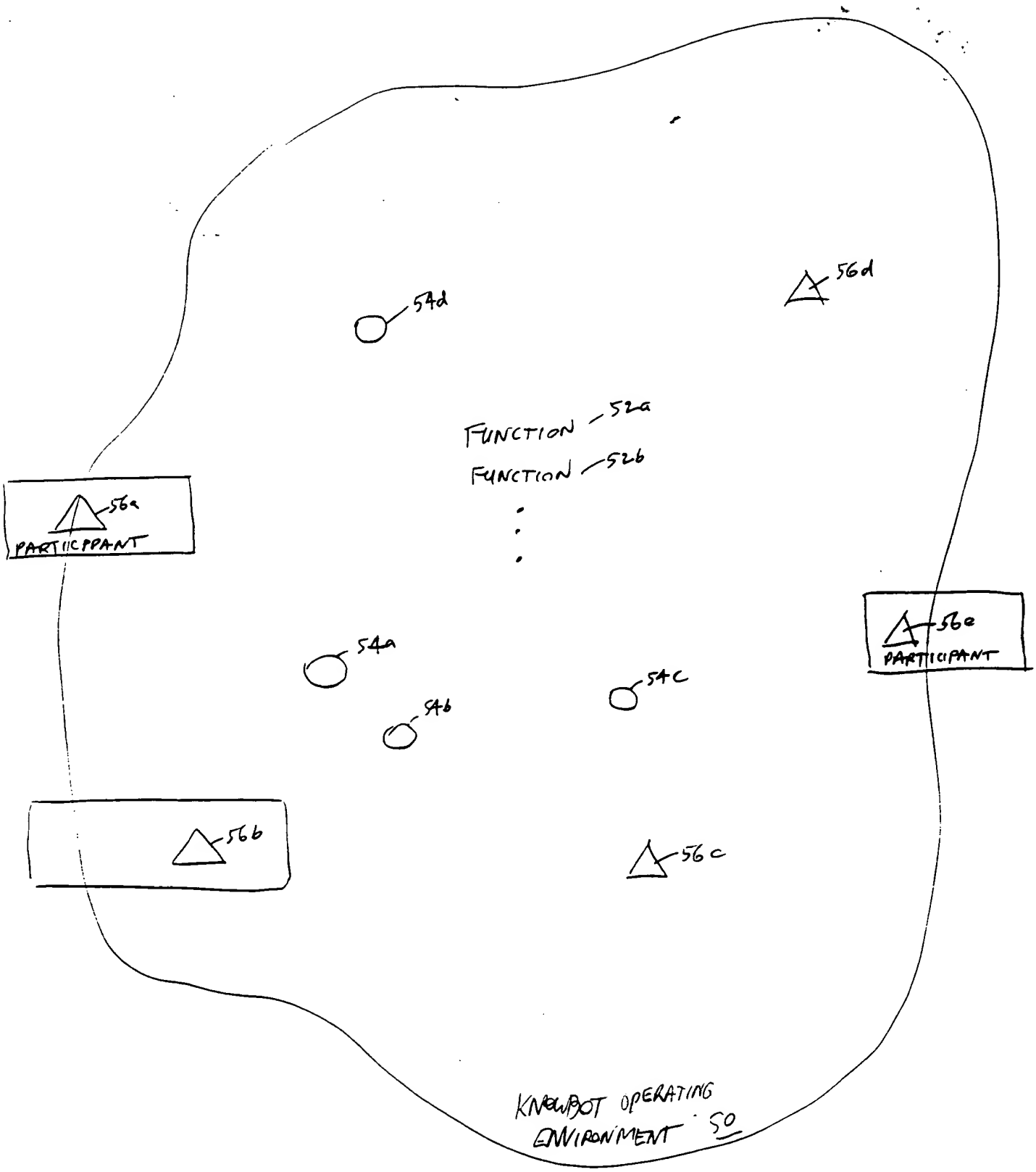


FIGURE 3

08720092 260294

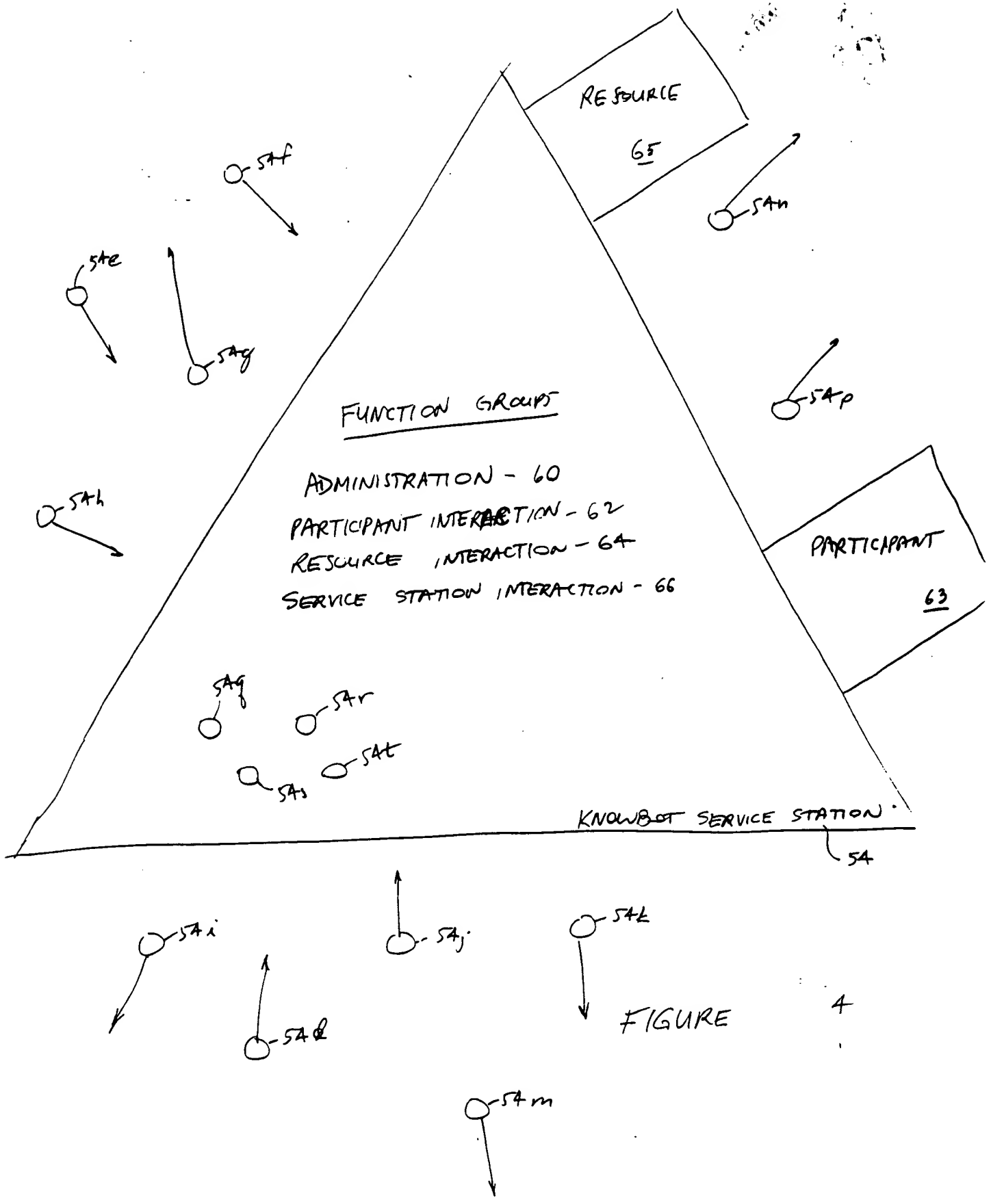


FIGURE 4

SUBSTITUTE SHEET (RULE 26)

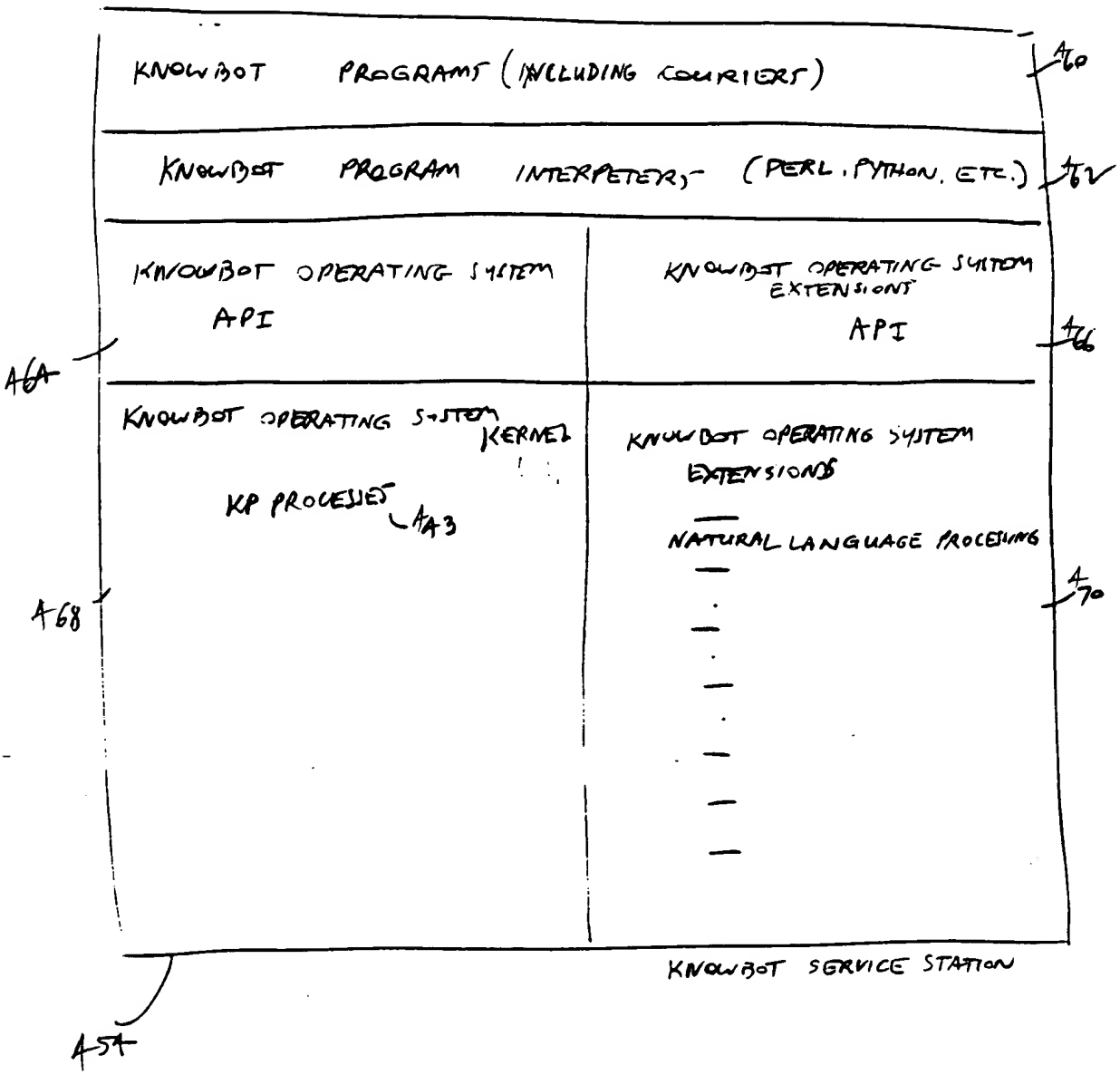
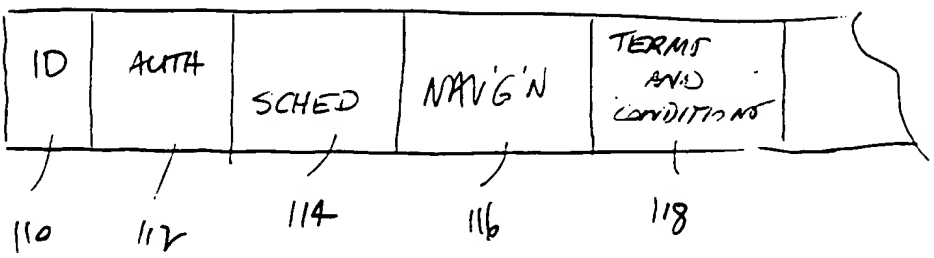

KNOWBOT PROGRAMS (INCLUDING COURIERS)
460
KNOWBOT PROGRAM INTERPETERS (PERL, PYTHON, ETC.)
462
KNOWBOT OPERATING SYSTEM API
KNOWBOT OPERATING SYSTEM EXTENSIONS API
466
464
KNOWBOT OPERATING SYSTEM KERNEL
KP PROCESSES
443
KNOWBOT OPERATING SYSTEM EXTENSIONS
NATURAL LANGUAGE PROCESSING
470
468
KNOWBOT SERVICE STATION
454

FIGURE 5



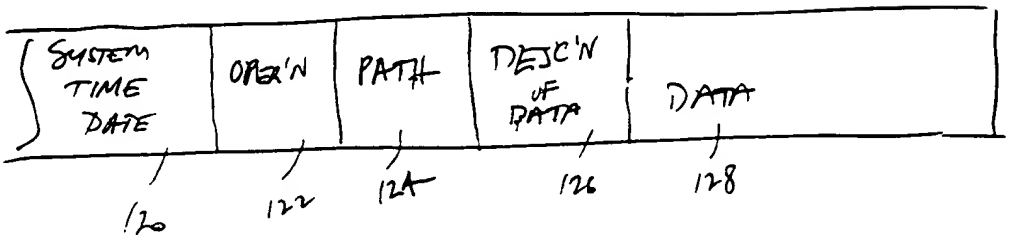


FIGURE 6

08720092 - 960208

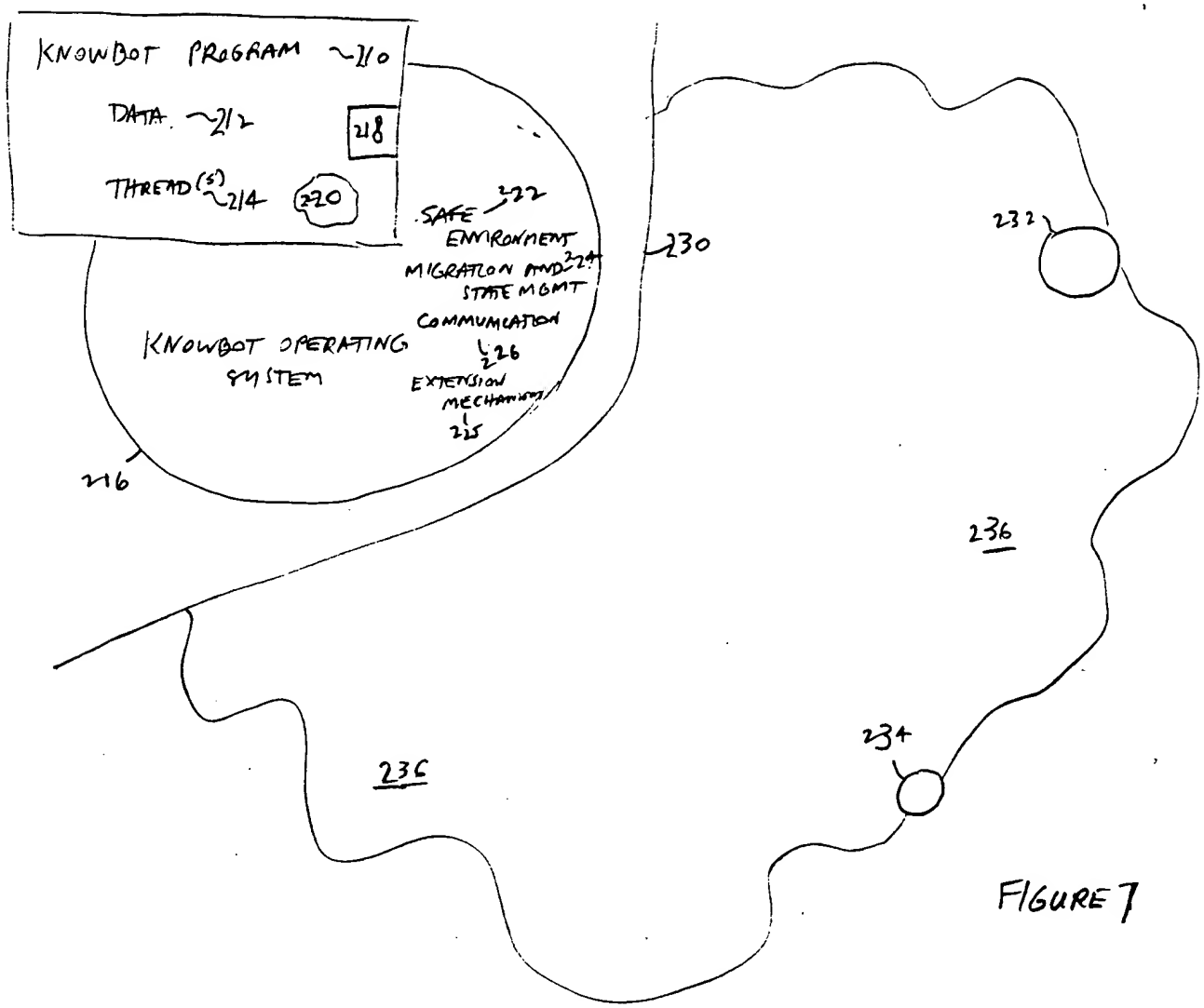

KNOWBOT PROGRAM 210
DATA 212
218
THREAD(S) 214
220
SAFE 222
ENVIRONMENT
MIGRATION AND 224
STATE MGMT
COMMUNICATION 226
EXTENSION MECHANISM 225
KNOWBOT OPERATING SYSTEM
216
230
232
234
236
236

FIGURE 7

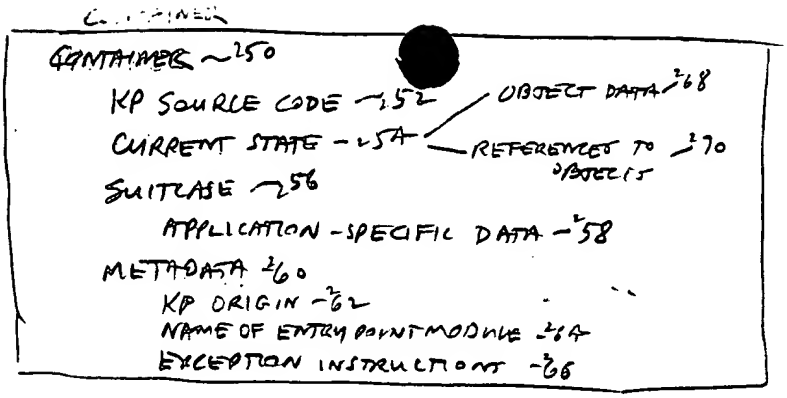


FIGURE 8

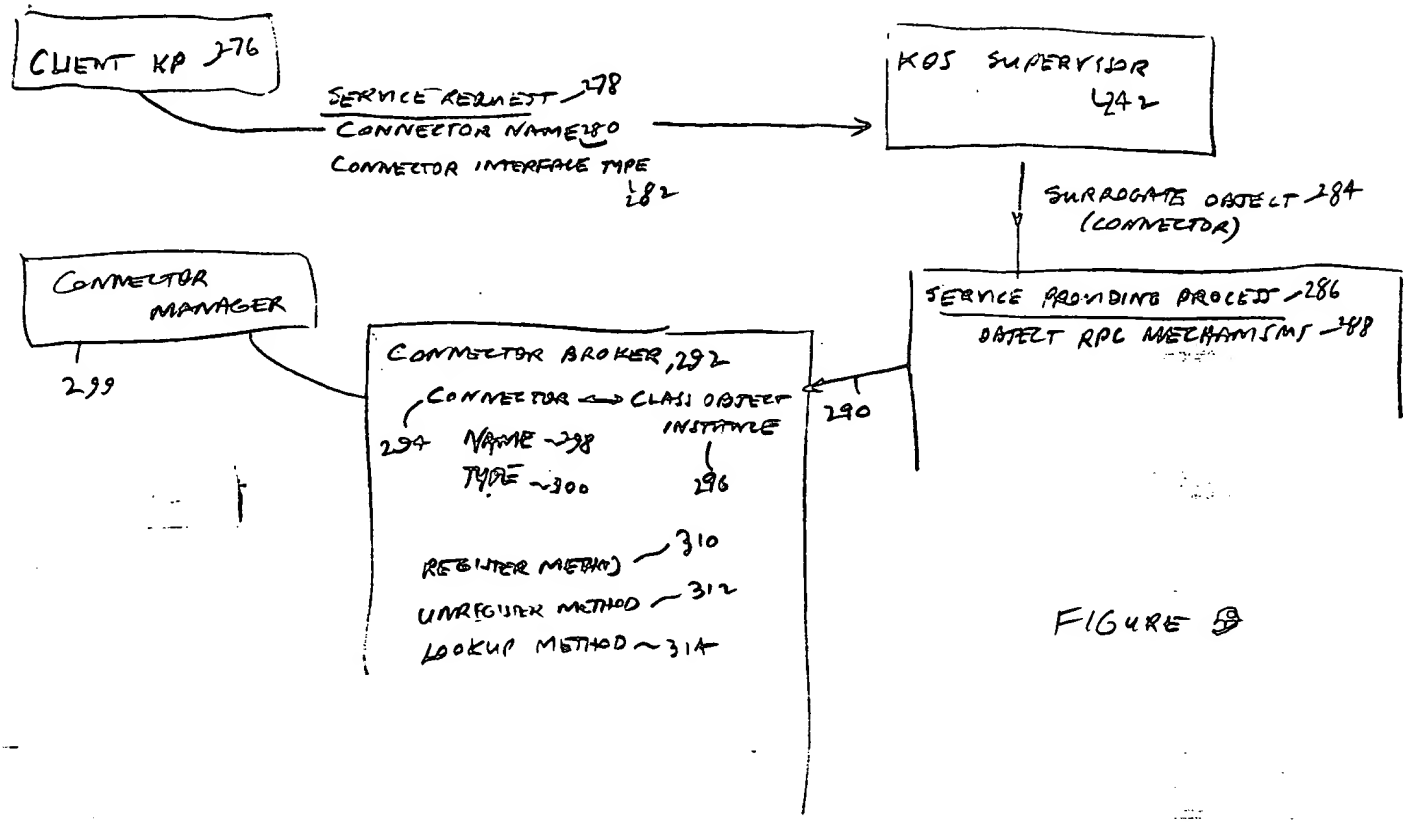


FIGURE 9

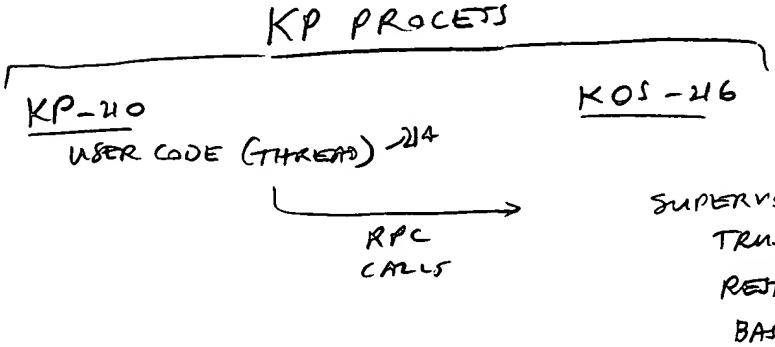


FIGURE 10

Docket No. 50277-0596

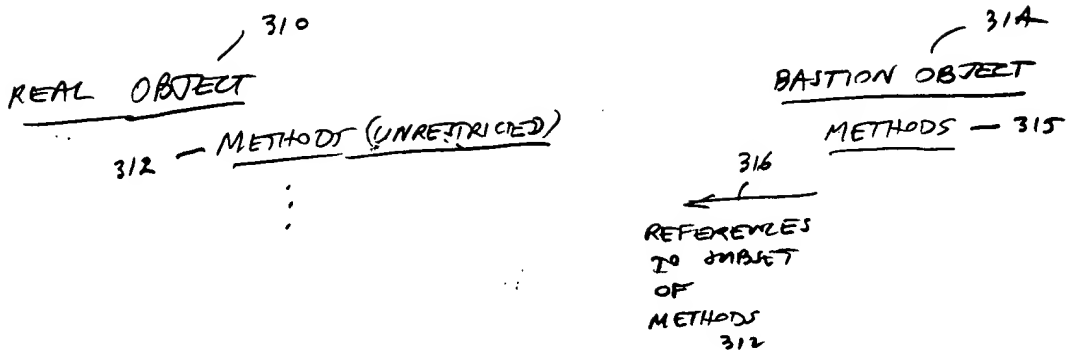


FIGURE 11

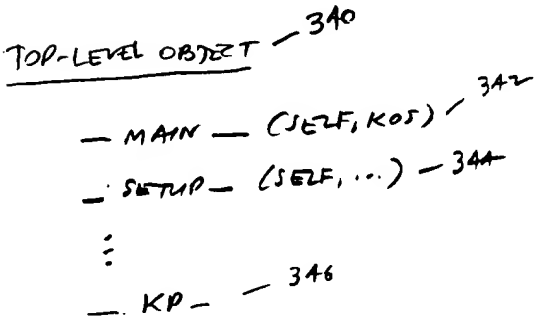


FIGURE 12

SEARCH REPORT

```
import rand                          # Python random number module
import nstools                       # helper module for using KOS namespace

class KP:

    def __init__(self):
        "Initialize KP's instance variables."
        self.maxhops = 20
        self.hopcount = 0
        self.visited = []            # list of KOSes that have been visited

    def __main__(self, kos):
        "Finds services available here, then migrates to a new KOS."
        self.find_services(kos, 'Search.Boolean')
        self.visited.append(kos.get_kos_name())
        self.hopcount = self.hopcount + 1
        if self.hopcount < self.maxhops:
            places = self.get_new_places(kos)
            if places:
                kos.migrate(rand.choice(places))

    def find_services(self, kos, service_type):
        "Save a list of available services in the suitcase"
        services = kos.list_services(service_type)
        file = kos.get_suitcase().open(kos.get_kos_name(), 'w')
        for serv in services:
            file.write(serv.name + '\n')
        file.close()

    def get_new_places(self, kos):
        "Return list of KOSes that have not been visited."
        descriptor = nstools.Lookup(kos.get_namespace(), 'world/kos')
        context = descriptor.Open('Namespace.Context')
        places = []
        for place in context.List():
            if place not in self.visited:
                places.append(place)
        return places
```

FIGURE 13

SUBSTITUTE SHEET (RULE 26)

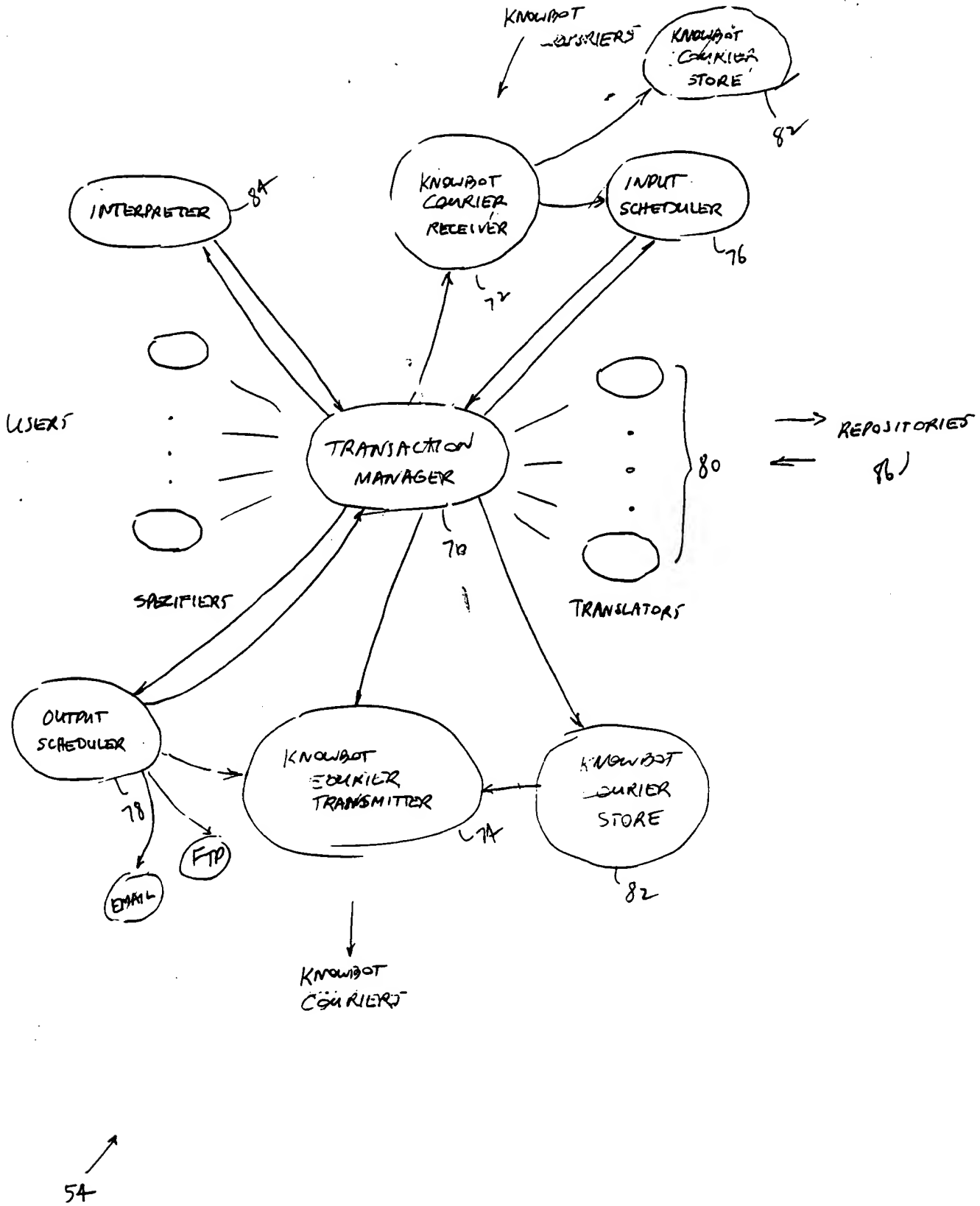

KNOWBOT COURIERS
KNOWBOT COURIER STORE
82
INTERPRETER
84
KNOWBOT COURIER RECEIVER
72
INPUT SCHEDULER
76
USERS
TRANSACTION MANAGER
70
80
REPOSITORIES
86
SANITIFIERS
TRANSLATORS
OUTPUT SCHEDULER
78
FTP
EMAIL
KNOWBOT COURIER TRANSMITTER
74
KNOWBOT COURIER STORE
82
KNOWBOT COURIERS
54

FIGURE 1A